SOFT MAIL

เปลี่ยนไดเรคทอรี เร็วกว่า NCD

อัลกอริทึมในการเขียนโปรแกรมเปลี่ยนไดเรคทอรีให้เร็วกว่า NCD พร้อมแนวทางการนำไปใช้งาน

ในการเปลี่ยนไดเรคทอรีด้วยคำสั่ง CD (Change Directory) ของดอสถ้าเป็นแผ่นดิสเกตต์ธรรมดาที่มีจำนวนไดเรคทอรีน้อย ๆ ก็คงจะไม่ยุ่งยากเท่าใดนัก แต่ถ้าไดเรคทอรีที่เราต้องการเข้าไปทำงานนั้น เป็นไดเรคทอรีอยู่ที่คนละกิ่งก้านสาขากันกับไดเรคทอรีปัจจุบันก็คงจะไม่ง่ายนัก จึงจำเป็นที่จะต้องมีโปรแกรมที่เข้ามาช่วยให้เกิดความสะดวกและรวดเร็วขึ้น โปรแกรมดังกล่าวนี้ที่รู้จักกันดีและกลายเป็นสิ่งจำเป็นที่ผู้ใช้งานฮาร์ดดิสก์จะขาดเสียไม่ได้ ก็คือโปรแกรม NCD.EXE ในนอร์ตันยูทิลิตี้ ซึ่งได้รับการยอมรับว่าทำงานได้รวดเร็วมาก แต่ต่อไปนี้เราสามารถที่จะเปลี่ยนไดเรคทอรีได้เร็วกว่า NCD เสียอีก

FCD ชื่อพระเอกของเรา

ใช่แล้วครับนั่นคือชื่อของโปรแกรมที่ผู้เขียนได้อวดสรรพคุณเอาไว้ข้างต้น แรงบันดาลใจที่ทำให้ผู้เขียนคิดสร้างโปรแกรมนี้ขึ้นมา ก็เนื่องจากผู้เขียนไปได้ยินคำกล่าวชมโปรแกรม NCD ของผู้ใช้หลาย ๆ ท่านว่า เป็นสุดยอดแห่งความเร็วในการเปลี่ยนไดเรคทอรี เพราะ NCD ใช้วิธีการเก็บ TREEINFO ทำให้เกิดคำถามกับตัวเองว่า น่าจะมีวิธีที่เร็วกว่านี้แน่ ๆ จึงได้ทดลองคิดอัลกอริทึมอันหนึ่งขึ้นมา ในครั้งแรก ผู้เขียนยังไม่ได้ค้นหาไดเรคทอรีทั้งหมดเก็บไว้เป็นไฟล์ TREEINFO เหมือนของ NCD พบว่ายังช้ากว่า NCD อยู่ จึงเพิ่มอัลกอริทึมสำหรับการสร้างและใช้งานไฟล์ TREEINFO ขึ้นมา ปรากฏว่าคราวนี้สามารถทำงานได้เร็วกว่า NCD และไฟล์ TREEINFO ของ FCD มีขนาดเล็กกว่าของ NCD มาก

เร็วกว่าไม่ได้หมายความว่าดีกว่า

เพราะ FCD ทำงานเฉพาะในรูปแบบของการป้อนชื่อไดเรคทอรีเป็นพารามิเตอร์ที่บรรทัดคำสั่งโดยตรง ซึ่งเป็นแบบที่เราใช้กันบ่อยกว่า แต่ NCD ยังสามารถทำงานในแบบเมนู ซึ่งแสดงโครงสร้างของไดเรคทอรีเป็นแบบต้นไม้ให้เลือกเปลี่ยนไดเรคทอรีได้ด้วย ยังไม่นับรวมถึงงานอื่น ๆ ที่ NCD สามารถทำได้อีก เช่น การ Prune & Graft หรือการทำสำเนาไดเรคทอรี ซึ่งต้องยอมรับว่ายอดเยี่ยมจริง ๆ

วิธีการ

คุณค่าของโปรแกรม FCD นี้ไม่ได้อยู่ที่เป็นตัวเลือกอันหนึ่งสำหรับการเปลี่ยนไดเรคทอรี แบบป้อนชื่อไดเรคทอรีที่บรรทัดคำสั่งให้เร็วกว่าเท่านั้น แต่ยังอยู่ที่อัลกอริทึมของโปรแกรมที่ผู้เขียนคิดว่า น่าจะเป็นประโยชน์ที่จะศึกษาด้วย จริงอยู่ ยังมีอัลกอริทึมที่สามารถทำงานได้เร็วกว่านี้อีก แต่จะมีความยุ่งยากในการเขียนโปรแกรมมากกว่า

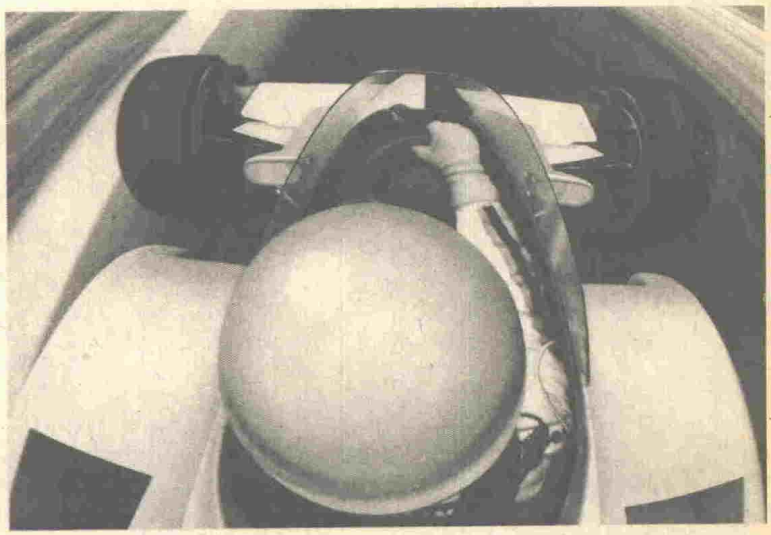
เรามาย้อนรอยกันดีกว่าว่าโปรแกรมนี้เกิดขึ้นมาได้อย่างไร โดยผู้เขียนจะกล่าวเน้นในหลักการ จะไม่ลงในรายละเอียดของซอร์สโค้ดของโปรแกรมมากนัก เนื่องจากโปรแกรมนี้จะเขียนด้วยภาษาอะไรก็ได้ที่สามารถเรียกใช้งานดอสอินเตอร์รัพต์ได้ และโปรแกรมมีขนาดค่อนข้างใหญ่จึงไม่สะดวกที่จะนำมาลงทั้งหมด ผู้ที่สนใจซอร์สโค้ดของโปรแกรม พร้อมทั้งโปรแกรมที่คอมพายล์แล้ว สามารถใช้บริการซอฟต์เมล์ได้เช่นเคย

ออกแบบการทำงาน

ก่อนอื่น เราต้องสรุปออกมาคร่าว ๆ เสียก่อนว่าเราจะให้โปรแกรมทำงานอะไรบ้าง งานที่ FCD จะทำ คือ

1. ถ้าชื่อไดเรคทอรีที่ต้องการเปลี่ยนไม่ใช่ไดเรคทอรีปัจจุบัน เมื่อค้นพบจะทำการเปลี่ยนไปยังไดเรคทอรีนั้นทันที แต่ถ้าชื่อตรงกันกับไดเรคทอรีปัจจุบันอยู่แล้วจะทำการค้นหาว่ามีไดเรคทอรีอื่นที่ชื่อซ้ำกันอยู่หรือไม่ด้วย

2. ถ้าไม่มีไดเรคทอรีที่ต้องการเปลี่ยนไปอยู่ จะเปลี่ยนไปยังไดเรคทอรีที่มีชื่อใกล้เคียงที่สุด คือ จำนวนตัวอักษรข้างหน้าที่เหมือนกันติดต่อกันมากที่สุดแทน (ถ้ามี)



3. ในการค้นหา ถ้ามีไฟล์ TREEINFO อยู่แล้วจะทำการค้นหาจาก TREEINFO แต่ถ้าไม่มีไฟล์ TREEINFO อยู่หรือถูกสั่งให้ทำการสร้างไฟล์ TREEINFO ใหม่ จะค้นหาจากส่วนไดเรคทอรีเซกเตอร์ (ด้วยดอสอินเตอร์รัพต์) โดยตรง

4. โปรแกรมจะต้องทำการเปลี่ยนไดเรคทอรีไปยังไดเรคทอรีที่ถูกซ่อนเอาไว้ได้

5. สามารถทำการสร้างไฟล์ TREEINFO ใหม่ได้ เมื่อมีการเปลี่ยนโครงสร้างของไดเรคทอรี โดยใช้พารามิเตอร์ /R

6. สร้างไฟล์ TREEINFO เฉพาะในฮาร์ดดิสค์เท่านั้น เนื่องจากเนื้อที่ของฮาร์ดดิสค์มีมาก การเก็บไฟล์ TREEINFO ซึ่งมีขนาดเล็กเข้าไปจะไม่ทำให้สิ้นเปลืองเนื้อที่ไปมาก นอกจากนี้ ฮาร์ดดิสค์จะถูกใช้งานบ่อยครั้ง การเก็บชื่อไดเรคทอรีทั้งหมดเป็นไฟล์ไว้ จะทำให้ไม่ต้องค้นหาจากไดเรคทอรีเซกเตอร์โดยตรงทุกครั้ง

7. เมื่อทำการสร้างไฟล์ TREEINFO ใหม่ ถ้าไฟล์เดิมถูกกำหนดให้มีลักษณะอ่านได้อย่างเดียว ไฟล์ใหม่จะถูกกำหนดด้วยลักษณะเดียวกัน โดยก่อนที่จะทำการเขียนใหม่จะต้องยกเลิกลักษณะการอ่านอย่างเดียวไว้ชั่วคราว

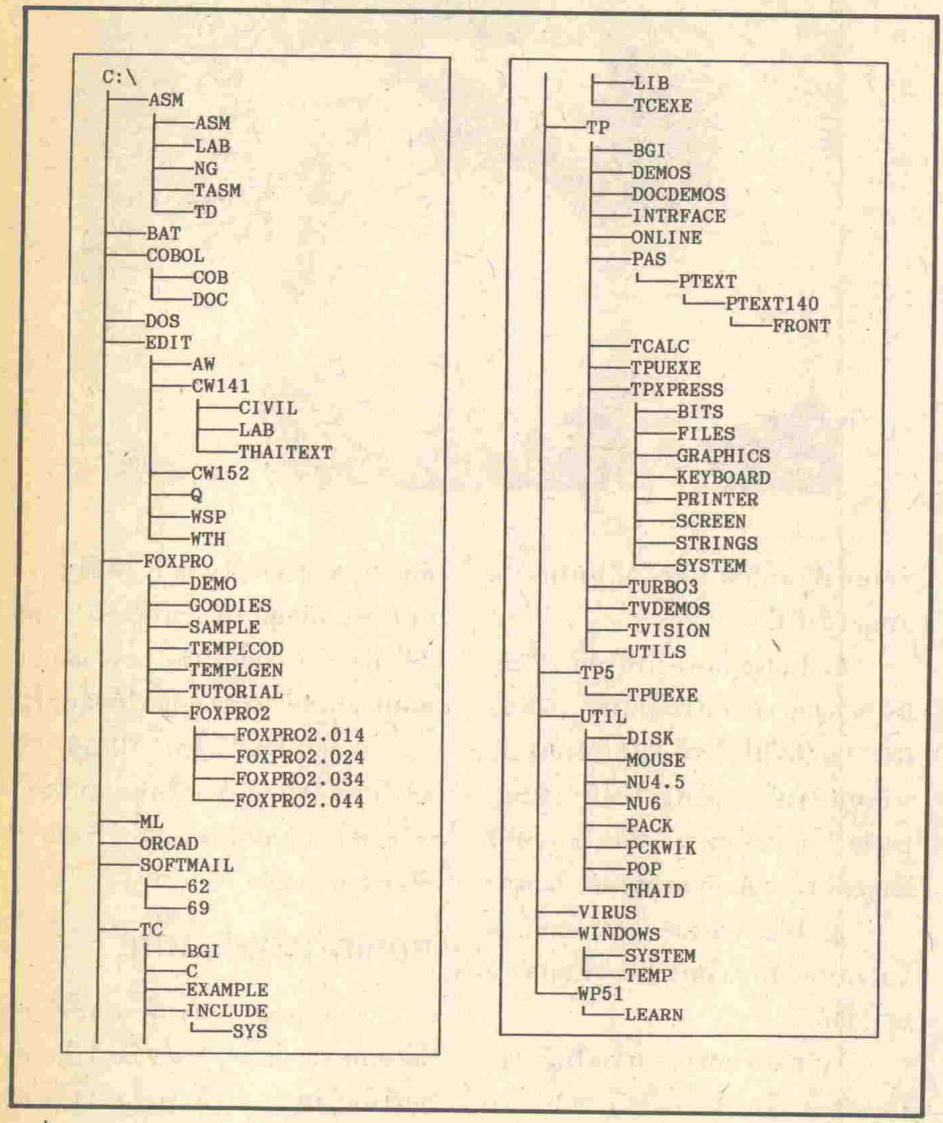
8. ก่อนใช้งานไฟล์ TREEINFO จะทำการตรวจสอบว่าไฟล์ถูกเปลี่ยนแปลงหรือไม่แต่เพียงคร่าว ๆ เพื่อให้เกิดความรวดเร็ว

ออกแบบอัลกอริทึม

1. เนื่องจากโดยปกติ เราจะใช้งานไดเรคทอรีที่อยู่ตื้น ๆ บ่อยครั้งกว่าไดเรคทอรีที่อยู่ลึก ๆ การค้นหาจึงควรค้นหาแบบค้นหาในทางกว้างก่อน (Breadth-first search)

2. ในระหว่างการค้นหา จะเก็บชื่อไดเรคทอรีไว้ในฮีพ เนื่องจากฮีพมีขนาดใหญ่และเราไม่รู้ขนาดของโครงสร้างไดเรคทอรีที่แน่นอน การใช้หน่วยความจำจึงควรใช้เท่าที่จำเป็น (อย่างไรก็ตาม โปรแกรม FCD ก็ยังใช้หน่วยความจำมากกว่าที่จำเป็นอยู่หลายส่วนทั้งนี้ เพื่อให้เกิดความรวดเร็ว)

3. จากโครงสร้างของไดเรคทอรีในรูปที่ 1 จะสร้างไฟล์ TREEINFO ได้เป็นดังในรูปที่ 2 ซึ่งอธิบายได้ดังนี้



รูปที่ 1

```
(ASM,BAT,COBOL,DOS,EDIT,FOXPRO,ML,ORCAD,SOFTMAIL,TC,TP,
TP5,UTIL,VIRUS,WINDOWS,WP51)(ASM,LAB,NG,TASM,TD)()(COB,
DOC)()(AW,CW141,CW152,Q,WSP,WTH)(DEMO,GOODIES,SAMPLE,TE
MPLCOD,TEMPLGEN,TUTORIAL,FOXPRO2)()()(62,69)(BGI,C,EXAM
PLE,INCLUDE,LIB,TCEXE)(BGI,DEMOS,DOCDEMOS,INTRFACE,ONLI
NE,PAS,TCALC,TPUEXE,TPXPRESS,TURBO3,TVDEMOS,TVISION,UTI
LS)(TPUEXE)(DISK,MOUSE,NU4.5,NU6,PACK,PCKWIK,POP,THAID)
()(SYSTEM,TEMP)(LEARN)()()()()()()()()(CIVIL,LAB,THAITE
XT)()()()()()()()()()()(FOXPRO2.014,FOXPRO2.024,FOXPRO2
.034,FOXPRO2.044)()()()()()(SYS)()()()()()()()()(PTEXT)()
()(BITS,FILES,GRAPHICS,KEYBOARD,PRINTER,SCREEN,STRINGS,
SYSTEM)()()()()()()()()()()()()()()()()()()()()()()()()()
(PTEXT140)()()()()()()()()(FRONT)()
```

รูปที่ 2

- วงเล็บแรก จะเป็นไดเรคทอรีทั้งหมด ในไดเรคทอรีมาก

- วงเล็บถัดมา จะเป็นไดเรคทอรีย่อยของไดเรคทอรีแรกในไดเรคทอรีราก

- วงเล็บถัดมาอีก จะเป็นไดเรคทอรีย่อยของไดเรคทอรีที่สองในไดเรคทอรีราก

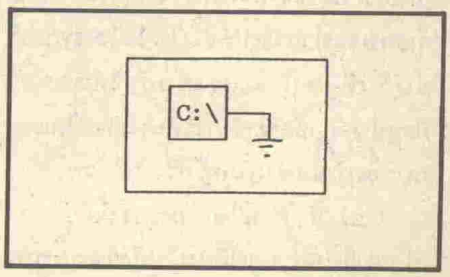
- การไม่มีไดเรคทอรีย่อยต่อไปอีก จะแทนด้วยวงเล็บว่าง

เช่น จากข้อมูลตัวแรกในวงเล็บที่ 1 คือไดเรคทอรี ASM ที่ราก จะมีไดเรคทอรีย่อยในชั้นถัดไปเป็น ASM, LAB, NG, TASM และ TD ตามลำดับ

- เป็นเช่นนี้ไปเรื่อย ๆ จนครบทุกไดเรคทอรี

4. ในการหาตำแหน่งของชื่อไดเรคทอรีที่ต้องการจะเปลี่ยนไปหา จาก TREEINFO จะทำการอ่านข้อมูลจากไฟล์เข้ามาเก็บในหน่วยความจำก่อน เพื่อให้เกิดความรวดเร็วในการค้นหา และการอ่านไฟล์จะทำการอ่านเป็นบล็อค เพื่อให้เกิดความรวดเร็วที่สุด นอกจากนี้หน่วยความจำที่ใช้เก็บ จะถูกเตรียมไว้ในสแต็กอย่างเพียงพอเอาไว้เป็นอะเรย์ เนื่องจากการจัดการกับอะเรย์จะรวดเร็วกว่าการจัดการกับตัวแปรในฮีพ

5. ในการค้นหาจาก TREEINFO จะเลียนแบบการทำงานของ FindFirst (ดอสอินเตอร์รัพต์ ฟังก์ชัน 11h) และ FindNext (ฟังก์ชัน 12h) โดยมีการ



รูปที่ 3

คืนค่าสถานะของการทำงานว่าค้นหาพบหรือไม่ ด้วยโค้ดเดียวกันกับดอสอินเตอร์รัพต์ เพื่อให้สามารถใช้ส่วนของโปรแกรมร่วมกันกับการค้นหาแบบปกติด้วยดอสอินเตอร์รัพต์ได้

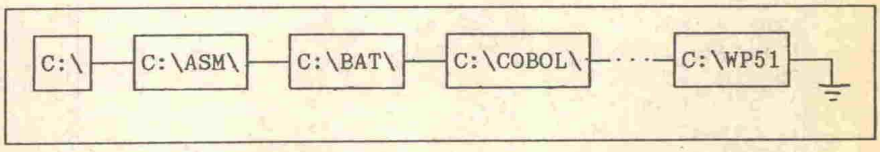
6. อัลกอริทึมในการค้นหาทุกไดเรคทอรีบนดิสค์ ไม่ว่าจะเป็นการค้นหาจาก TREEINFO หรือด้วยดอสอินเตอร์รัพต์ (ในเทอร์โบปาสดอส คือ FindFirst และ FindNext) จะใช้วิธีเดียวกัน คือการค้นหาโดยใช้ลิงค์ลิสต์เข้ามาช่วย

– เริ่มแรก กำหนดโหนดแรกของลิงค์ลิสต์ให้เก็บชื่อไดเรคทอรีรากเอาไว้

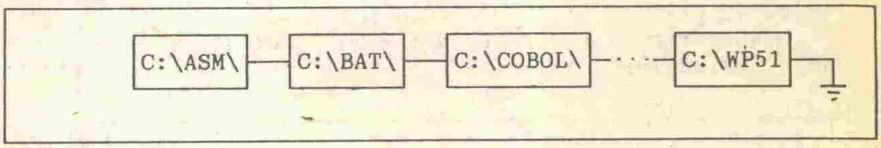
– เริ่มการค้นหาไดเรคทอรีย่อยที่โหนดแรก (ไดเรคทอรีราก) เมื่อพบไดเรคทอรีย่อยจะทำการสร้างโหนดใหม่เก็บต่อท้ายไปเรื่อย ๆ ถ้าไม่พบจะทำการยกเลิกโหนดที่กำลังค้นหา แล้วทำการค้นหาโหนดต่อ ๆ ไปเรื่อย ๆ จนกระทั่งไม่สามารถค้นหาต่อไปได้อีก (จะเห็นว่าสามารถกำหนดให้โปรแกรมย่อยส่วนนี้ทำงานแบบเรียกตัวเอง หรือเป็นลูปได้)

หากจะกล่าวไว้เพียงลอย ๆ เช่นนี้คงจะยังมองไม่เห็นภาพ จะขอยกตัวอย่างจากโครงสร้างของไดเรคทอรีในรูปที่ 1 ดังนี้

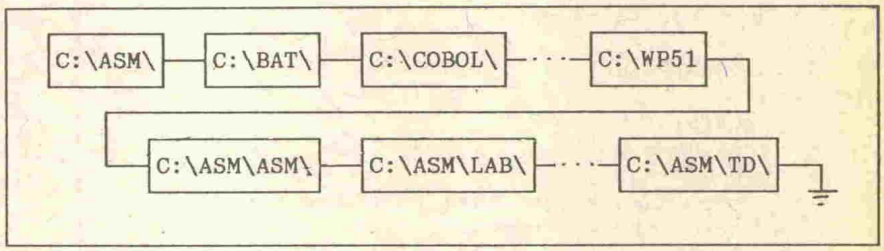
– เริ่มแรก เราจะสร้างโหนดแรกดังรูปที่ 3 (สมมติว่าโครงสร้างดังกล่าวอยู่ในไดรฟ์ C)



รูปที่ 4



รูปที่ 5



รูปที่ 6

– เริ่มทำการค้นหาในโหนดแรกไปเรื่อย ๆ จะได้ลิงค์ลิสต์เป็นดังในรูปที่ 4

(ส่วนที่ละเอาไว้ คือไดเรคทอรีอื่น ๆ ที่อยู่ระหว่าง ML กับ WP51)

– จากรูปที่ 4 เมื่อทำการค้นหาต่อไปจะไม่พบไดเรคทอรีย่อยของ C:\ ซึ่งเป็นโหนดแรกอีกต่อไป จะทำการยกเลิกโหนดแรก ดังรูปที่ 5 แล้วทำการค้นหาในโหนดแรกอันใหม่ต่อไปแทนดังรูปที่ 6

– ทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งไม่สามารถทำการค้นหาไดเรคทอรีย่อยต่อไปได้อีก

เขียนเป็นโปรแกรม

เนื่องจากซอร์สโปรแกรมมีขนาดค่อนข้างใหญ่ (แต่โปรแกรมที่คอมพายล์แล้วไม่ใหญ่มากนะครับ) จึงขอกล่าวถึงเฉพาะส่วนของโปรแกรมที่เป็นอัลกอริทึมที่สำคัญของโปรแกรมนี้ 3 ส่วน คือ FindFirst2 และ FindNext2 (ที่เลียนแบบการทำงานของดอสอินเตอร์รัพต์) สำหรับการค้นหาไดเรคทอรีจาก TREEINFO และ CheckNode สำหรับการค้นหาไดเรคทอรีย่อยทุก ๆ ไดเรคทอรี โดยจะเป็นดังในรูปที่ 7, 8 และ 9 ตามลำดับ ซึ่งอธิบายการทำงานได้ดังนี้

```
Procedure  FindFirst2;
{ FindFirst for InfoMem }
begin
  Point:= Point + 2;
  if InfoMem[Point] <> ')' then
    begin
      srec.name:= '';
      repeat
        srec.name:= srec.name + InfoMem[Point];
        srec.attr:= Directory;
        inc(Point);
      until InfoMem[Point] in [',',')'];
      DosError:= 0;
    end
  else
    DosError:= 2; { Directory ( File ) not found }
end;  { FindFirst2 }
```

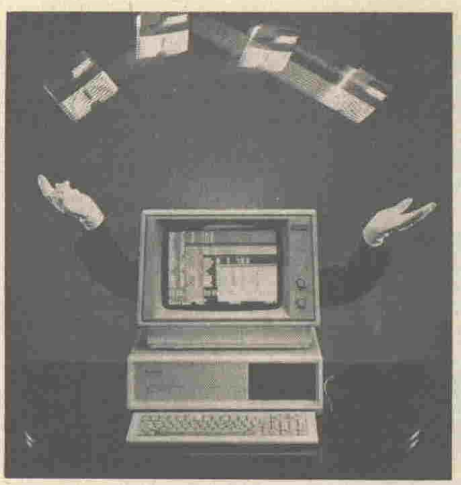
รูปที่ 7

```
Procedure  FindNext2;
{ FindNext for InfoMem }
begin
  if InfoMem[Point] <> ')' then
    begin
      inc(Point);
      srec.name:= '';
      repeat
        srec.name:= srec.name + InfoMem[Point];
        srec.attr:= Directory;
        inc(Point);
      until InfoMem[Point] in [',',')'];
      DosError:= 0;
    end
  else
    DosError:= 2; { Directory ( File ) not found }
end;  { FindNext2 }
```

รูปที่ 8

```
begin  { CheckNode }
  if node <> nil then
    begin
      FindSubDOfD(node^.dir);
      if (not FoundDir) or (NewInfoFileOpened) then
        begin
          OldNode:= node;
          node:= node^.link;
          CheckNode(node);
        end;
    end
  else Dispose(OldNode);
end;  { CheckNode }
```

รูปที่ 9



FindFirst2 และ FindNext2

ขั้นตอนในการค้นหาด้วย FindFirst2 และ FindNext2 จะมองย้อนกลับกับตอนสร้าง TREEINFO คือเริ่ม FindFirst2 หลังจากเครื่องหมายวงเล็บเปิดใน TREEINFO และ FindNext2 ไปเรื่อย ๆ จนพบเครื่องหมายวงเล็บปิด จะคืนค่า DosError กลับมาเป็น 2 ซึ่งหมายถึง ไม่พบไดเรคทอรีย่อยของไดเรคทอรีที่กำลังทำการค้นหาอีกต่อไป

CheckNode

สำหรับการทำงานของโปรแกรมย่อยแบบเรียกตนเองส่วนนี้ ได้อธิบายไว้ละเอียดแล้วในขั้นที่สอง (ออกแบบอัลกอริทึม) ในข้อ 6

ส่งท้าย

จากวิธีดังกล่าวข้างต้น ยังสามารถนำไปประยุกต์ใช้ในกรณีอื่น ๆ ได้อีก เช่นการค้นหาไฟล์, การลบไฟล์ .BAK ในทุกไดเรคทอรี, การค้นหาไวรัสในไฟล์ .SYS .COM และ .EXE ในทุกไดเรคทอรี เป็นต้น หวังว่าโปรแกรมนี้คงจะเป็นประโยชน์ต่อผู้อ่านบ้าง ไม่มากก็น้อย